In the United States Patent and Trademark Office

Serial number 09/340,172
Application filed June 25, 1999
Applicant Derek Wong
Application Title Methods for Increasing Instruction-Level Parallelism in Microprocessors and Digital Systems
Examiner/GAU Eric Coleman / 2183

Mailed December 12, 2002
at San Jose, California

From:
Derek Wong
1341 Echo Valley Dr.
San Jose, CA 95120

To:
Assistant Commissioner for Patents
United States Patent and Trademark Office
Washington, DC 20231

RE: Amendment B to patent application 09/340,172

Thank you for reviewing patent application 09/340,172.

In response to the office action letter regarding patent application 09/340,172 mailed on June 12, 2002, the applicant encloses an amended claim set and this letter.

Claims 44-77 are unchanged. Claims 78-85 have been amended. Claim 86 has been cancelled. The applicant has made effort to respond to each of the comments received in the office action below.

1. Claim 78 has been amended. The mechanisms in Yeh (Patent 5,903,750) provide methods of branch prediction. The "predicated" instructions in Yeh are limited to conditional branch instructions. Yeh does not describe any method to deal with other kinds of conditionally-executed instructions. The present patent application describes use of a predicate history table to

help with the scheduling of predicated instructions other than conditional branch instructions (see page 47 of specification). Therefore, the amended claim 78 has been amended with the limitation that the claimed use of a predicate history table to help with scheduling is for an instruction set architecture that has predicated instructions other than conditional branch instructions.

2. Claim 79 has been amended. Kumar (US Patent 6,247,094) describes a technique where a hit/miss prediction and way prediction is made during the access to a memory address. These predictions influence the steps of accessing the memory address through the cache hierarchy and main memory. The office action indicates that this could be used to help optimize subsequent instruction scheduling although Kumar does not detail this explicitly. However, Kumar's technique only provides predictions about a particular memory access after the access is started, i.e. the prediction is provided after the execution of the memory access has started. Claim 79 has been amended with the limitation that the hit/miss history table is used in conjunction with scheduling instructions prior to executing the instructions. For a memory access instruction, this means scheduling the instruction prior to beginning the memory access. Kumar's technique would not be able to help do such scheduling.

3. Claim 80 has been amended to be a dependent claim upon claim 76 which has been allowed. Hence, amended claim 80 should be allowable subject matter.

4. Claim 81 has been amended. Farber (US Patent 6,105,124) describes a software method of binary translation from a source instruction set architecture to a target instruction set architecture. The implementation described by Farber stores the translated blocks of instructions in memory. Glew (US Patent 5,948,097) describes a method of using physical registers and committing their results to logical registers during dynamic run-time speculation. The mapping

of physical to logical registers in Glew is done through an information table created dynamically by the execution unit. Instructions in the instruction set architectures used by Farber and Glew do not have any explicit notations to indicate static register renaming of logical registers to physical registers prior to execution, as described in the present patent application (pages 50-53). Farber does not have a description of renaming, and Glew uses a run-time information table to contain dynamic (rather than static) register renaming information (i.e. the instructions in Glew do not contain renaming information.) Therefore, the combination of Farber and Glew would not suggest a design that has static register renaming with notations in the transformed instruction set architecture to support this. Claim 81 has been amended with limitations to include the use of static register renaming with explicit notations in the transformed instruction set architecture.

5. Claim 82 has been amended. Shiell (US Patent 5,911,057) describes a method of using instruction sequence numbers during run-time execution. The claimed inventive idea in amended claim 82 is that instruction sequence numbers from the original instruction set architecture (ISA) instructions are used to explicitly tag the transformed ISA instructions so that later execution of the transformed ISA instructions can still be done in the original program order of the original ISA instructions. Farber does not indicate any method of keeping the execution of translated blocks to be in the exact same instruction order as source blocks or that such is a desirable goal. Shiell only describes a method of run-time execution using instruction sequence numbers generated on the fly. The combination of the two would not yield claim 82's idea of explicitly using instruction sequence numbers statically during the process of transformation from an original ISA to transformed ISA, so that any later execution of the transformed ISA instructions would be kept in the same order as the original ISA instructions.

6. Claim 83 has been amended. Moshovos (US Patent 5,781,752) describes a method of handling speculative execution in a superscalar processor that is based in part on using a table containing potential data dependencies. In Moshovos, the table that is created is used for dynamic run-time scheduling. In contrast, amended claim 83 describes the use of a dependency matrix with a software method of scheduling for later execution.

7. Claim 84 has been amended. The intent of the claim is to claim an instruction set architecture that has fields to explicitly note dependencies between instructions using a dependency vector format. The cited reference Bharadwaj (US Patent 5,787,287) describes an idea called dependency path vector which is not part of the instruction set architecture. Bharadwaj describes the idea of a dependency path vector as used in a software compiler (col 1, lines 13-26). According to Bharadwaj (col 6, line 40 through col 9, line 22), dependency path vectors can be computed by a software compiler to help with scheduling program code. Bharadwaj does not make any suggestion that dependency path vectors should be placed into fields in the instruction set architecture of a processor. The amended claim clarifies the idea of having fields in an instruction set architecture to explicitly note dependencies using a dependency vector format.

8. Claim 85 has been amended. The intent of the claim is to claim an instruction set architecture that has fields to explicitly note dependencies between instructions using a dependency pointer format. Bharadwaj does not make any suggestion that dependency path vectors or pointers should be placed into any fields in the instruction set architecture of a processor. The amended claim clarifies the idea of having fields in an instruction set architecture to explicitly note dependencies using a dependency pointer format.